Session-based concurrency: between operational and declarative views

Cano Grijalba, Mauricio

DOI:
10.33612/diss.108552669

Publication date:
2020

Stellingen

behorende bij het proefschrift

Session-Based Concurrency: Between Operational and Declarative Views

van

Mauricio Alejandro Cano Grijalba

1. Session-based concurrency is a powerful verification technique for communication correctness in message-passing systems. While it appropriately expresses operational requirements, it does not often consider declarative requirements such as time and partial information. [Chapter 1]

2. Encodings are translations from terms of a source language into terms of a target language. They have been widely used to compare calculi for concurrency. Correct encodings can also be used to articulate a unified view for the analysis of message-passing systems. [Chapter 1]

3. Although there are many conceivable correctness criteria for encodings, one must be careful when selecting which ones are used because not all of them serve every purpose. [Chapter 3]

4. When relating operational and declarative languages three correctness criteria are indispensable: name invariance, compositionality, and operational correspondence (divided into completeness and soundness). [Part II, Part III]

5. Advanced session type systems can be a blessing and a curse: they can express complex communication patterns that appear in realistic settings; however, well-typed π-calculus processes implementing these patterns cannot always be compiled down into declarative specifications. [Chapter 3]

6. A unified view for the analysis of message-passing systems requires limiting session type systems when they are overly expressive. The resulting type systems can elegantly characterize sub-classes of well-typed π-calculus programs that can be correctly compiled into declarative specifications. [Chapter 3]

7. Partial information in concurrent systems can be appropriately represented by constraint-based languages. In particular, `lcc` is well-suited to enhance

session-based specifications with partial information. [Part II]

8. ReactiveML, a synchronous reactive language, can easily represent session-based concurrency by modeling communication channels as events associated to values. ReactiveML is well-suited to enhance session-based specifications by describing protocols with explicit reactions to events and time instants. [Part III]

9. In proving operational correspondence for encodings of operational languages into declarative languages such as `lcc` and ReactiveML, completeness proofs are often straightforward. Soundness proofs are much more challenging: one needs to establish invariants on the structure of translated terms to ensure that these terms correctly reflect the behavior of source terms. [Part II, Part III]

10. Encodings of π-calculi with binary sessions into ReactiveML follow an intuitive strategy that slices session synchronizations into time instants. This strategy does not scale up to multiparty sessions because synchronizations between multiple partners cannot be sliced into instants without inducing message loss. [Part IV]

11. The synchronous reactive features in Multiparty Reactive Sessions enable the analysis of session fidelity and communication correctness, but also allows us to address two new properties: output persistence and input timeliness. These properties ensure liveness for communication protocols in the presence of time instants and events. [Part IV]

12. Developing correct encodings is an art. It requires time and dedication. Nowadays, even art can be computer-generated. Hence, we must also think about the role that mechanizing the metatheory of programming languages could play when defining encodings and establishing their correctness.